Lógico 4: functores, generación, listas con orden

Ejercicio 1

En una empresa multinacional se da servicio de outsorcing a los servidores de varias empresas. De cada servidor se conoce su ubicación en una fila de servidores del Centro de Cómputos, y qué criticidad tiene asociada en el contrato de outsorcing. La criticidad (del 1 al 4) define qué tan rápido se tiene que dar solución a los problemas que se presentan en el server.
En el Centro de Monitoreo de la empresa se reciben las alertas de los problemas que acontence en el Centro de Cómputos. Los eventos que se informa son:

- cortes de luz que afectan toda una fila de servidores
- rebooteos de servers
- "cuelgues" de aplicaciones en un server.

Escribir las cláusulas que hagan falta para definir los predicados requiereAtencionNormal/2 y requiereAtencionInmediata/2, sabiendo que

- ante un corte de luz, todos los servidores de la fila requieren atención inmediata
- ante un rebooteo de server, el server afectado y los otros servidores que sean clientes de ese server requieren atención normal.
- ante un cuelgue de aplicación, el servidor afectado requiere atención inmediata si es crítico (criticidad 1 y 2), o atención normal si el servidor no es crítico (criticidad 3 y 4). Ningún otro servidor requiere atención.

Para los dos predicados, el primer argumento es el servidor del cual se quiere saber si requiere o no atención, y el segundo es un evento; usar functores para distinguir entre distintos eventos.
Lograr para ambos predicados la inversibilidad necesaria para que pueda consultarse qué servidores requieren atención dado un evento.

Los datos de los servidores se pueden modelar así:

```
servidor(PS1, fila1, criticidad1).
servidor(PS2, fila1, criticidad2).
servidor(WAS1, fila2, criticidad1).
servidor(WAS1_2, fila2, criticidad4).
servidor(FS_X48, fila2, criticidad1).
esCliente(PS1,FS_X48).   % acá dice que el servidor PS1 es cliente de FS_X48
esCliente(WAS1,FS_X48).  % acá dice que el servidor WAS1 es cliente de FS_X48
% etc...
```

En este ejemplo:

- ante un corte de luz en la fila1, los servidores PS1 y PS2 requieren atención inmediata.
- ante un rebooteo de FS_X48, tanto ese server como PS1 y WAS1 requieren atención normal.
- ante un cuelgue en PS2, ese server requiere atención inmediata y nadie más.
- ante un cuelgue en WAS1_2, ese server requiere atención normal y nadie más.

Ejercicio 2

La agencia matrimonial del ejercicio 3 de la práctica 1 arma grupos de mujeres y varones en igual cantidad. Para cada persona del grupo se determina, con los métodos científicos que usa la agencia, el orden de preferencia de las personas del otro sexo.

Damos un ejemplo de grupo:

- mujeres: Ana, Nora y Marta
- varones: Luis, Juan y Pedro
- preferencias de Ana : Luis / Juan / Pedro
- preferencias de Nora : Juan / Pedro / Luis
- preferencias de Marta : Pedro / Luis / Juan
- preferencias de Luis : Ana / Nora / Marta
- preferencias de Juan : Marta / Ana / Nora
- preferencias de Pedro : Nora / Marta / Ana

la agencia quiere armar parejas que incluyan a todas las personas de un grupo, y tal que el conjunto de parejas armadas sea estables.

Un conjunto de parejas es estable si todas sus parejas son estables en el contexto del conjunto.
Una pareja es inestable en un contexto si alguno de sus dos integrantes quiere dejar al otro. Una persona A quiere dejar a otra B si existe una tercera C tal que:

- A prefiere a C antes que a B, y
- C prefiere a A antes que a su pareja (la que tiene asignada en el contexto).

P.ej. el conjunto de Ana-Pedro, Nora-Luis, y Marta-Juan es inestable porque Ana quiere dejar a Pedro, dado que Ana prefiere a Luis antes que a Pedro, y Luis prefiere a Ana antes que a Nora (la pareja de Luis en el conjunto).

Se pide:

a. Definir el predicado parejas/3 tal que arma los conjuntos posibles de parejas a partir de un conjunto de mujeres y otro de varones. Hay que definir cómo representar una pareja.

b. Definir el predicado insatisfecho/2, que se verifica para una persona y un conjunto de parejas si la persona quiere dejar a su pareja asignada en el contexto del conjunto.
P.ej. Ana sí está insatisfecha respecto del conjunto de parejas del ejemplo.

c. Definir el predicado estable/1, que indique si un conjunto de parejas es estable o no. Hacerlo de forma tal de garantizar que sea inversible.
Se supone que las mujeres, los varones y los gustos de cada quién están en hechos, que (obviamente) hay que modelar.

Ejercicio 3

Se organiza un juego que consiste en ir buscando distintos objetos por el mundo. Cada participante está en un determinado nivel, cada nivel implica ciertas tareas, cada tarea consiste en buscar un objeto en una ciudad.

Representamos las tareas como functores buscar(Cosa,Ciudad), y definimos el predicado tarea/2 de esta forma:

```
tarea(basico,buscar(libro,jartum)).
tarea(basico,buscar(arbol,patras)).
tarea(basico,buscar(roca,telaviv)).
tarea(intermedio,buscar(arbol,sofia)).
tarea(intermedio,buscar(arbol,bucarest)).
tarea(avanzado,buscar(perro,bari)).
tarea(avanzado,buscar(flor,belgrado)).
```

o sea, si estoy en el nivel básico, mis tareas posibles son buscar un libro en Jartum, un árbol en Patras o una roca en Tel Aviv.

Para definir en qué nivel está cada participante se define el predicado nivelActual/2, de esta forma:

```
nivelActual(pepe,basico).
nivelActual(lucy,intermedio).
nivelActual(juancho,avanzado).
```

También vamos a necesitar saber qué idioma se habla en cada ciudad, qué idiomas habla cada persona, y el capital actual de cada persona. Esto lo representamos con los predicados idioma/2, habla/2 y capital/2:

```
idioma(alejandria,arabe).
idioma(jartum,arabe).
idioma(patras,griego).
idioma(telaviv,hebreo).
idioma(sofia,bulgaro).
idioma(bari,italiano).
idioma(bucarest,rumano).
idioma(belgrado,serbio).

habla(pepe,bulgaro).
habla(pepe,griego).
habla(pepe,italiano).
habla(juancho,arabe).
habla(juancho,griego).
habla(juancho,hebreo).
habla(lucy,griego).

capital(pepe,1200).
capital(lucy,3000).
capital(juancho,500).
```

Definir los siguientes predicados:
a. destinoPosible/2 e **idiomaUtil/2**.
destinoPosible/2 relaciona personas con ciudades; una ciudad es destino posible para un nivel si alguna tarea que tiene que hacer la persona (dado su nivel) se lleva a cabo en la ciudad. P.ej. los destinos posibles para Pepe son: Jartum, Patras y Tel Aviv.
idiomaUtil/2 relaciona niveles con idiomas: un idioma es útil para un nivel si en alguno de los destinos posibles para el nivel se habla el idioma. P.ej. los idiomas útiles para Pepe son: árabe, griego y hebreo.

b. excelenteCompaniero/2, que relaciona dos participantes. P2 es un excelente compañero para P1 si habla los idiomas de todos los destinos posibles del nivel donde está P1.
P.ej. Juancho es un excelente compañero para Pepe, porque habla todos los idiomas de los destinos posibles para el nivel de Pepe.
Asegurar que el predicado sea inversible para los dos parámetros.

c. interesante/1: un nivel es interesante si se cumple alguna de estas condiciones

- todas las cosas posibles para buscar en ese nivel están vivas (las cosas vivas en el ejemplo son: árbol, perro y flor)
- en alguno de los destinos posibles para el nivel se habla italiano.
- la suma del capital de los participantes de ese nivel es mayor a 10000

Asegurar que el predicado sea inversible.

d. complicado/1: un participante está complicado si: no habla ninguno de los idiomas de los destinos posibles para su nivel actual; está en un nivel distinto de básico y su capital es menor a 1500, o está en el nivel básico y su capital es menor a 500.

e. homogeneo/1: un nivel es homogéneo si en todas las opciones la cosa a buscar es la misma. En el ejemplo, el nivel intermedio es homogéneo, porque en las dos opciones el objeto a buscar es un árbol.
Asegurar que el predicado sea inversible.
Ayuda: hay que saber de alguna forma si una lista tiene un único elemento, p.ej. la lista [3,3,3,3] tiene un único elemento (el 3) mientras que la lista [2,1,2,4] no. Tal vez convenga definir un predicado aparte para este problema.

f. poliglota/1: una persona es políglota si habla al menos tres idiomas.

En general: es válido agregar los predicados necesarios para poder garantizar inversibilidad o auxiliares para resolver cada ítem, y usar en un ítem los predicados definidos para resolver ítems anteriores.

Ejercicio 4

Para un proyecto de desarrollo de software se tiene una base de conocimientos con los siguientes hechos:

tarea/3, que relaciona una tarea con su duración y un rol para realizarla.

```
tarea(login, 80, programador).
tarea(cacheDistribuida, 120, arquitecto).
tarea(pruebasPerformance, 100, tester).
tarea(tuning, 30, arquitecto).
```

predece/2, que relaciona dos tareas A y B indicando que la tarea B no puede comenzar hasta que la A no haya terminado.

```
precede(cacheDistribuida, pruebasPerformance).
precede(pruebasPerformance, tuning).
```

realizada/1, que indica las tareas que ya están terminadas.

```
realizada(login).
```

Se pide:

1. Definir el predicado anterior/2 que relaciona dos tareas A y B si estas deben realizarse en un orden específico, ya sea por una dependencia directa o indirecta. Por ejemplo:
?- anterior(T, tuning).
T = pruebasPerformance; (-> dependencia directa, porque pruebasPerformance precede a tuning)
T = cacheDistribuida; (-> dependencia indirecta, a través de pruebasPerformance)

2. Definir los predicados simple/1 y riesgo/1 de forma que:

- Las tareas que realizan los programadores y testers son consideradas simples. También se consideran simples todas las tareas que toman menos de una semana (40 horas).
- Consideramos como riesgos a todas las tareas de más de 40 horas que aún no han sido realizadas.

Por ejemplo:
```
?- simple(T).
T = login; (-> porque la realiza un programador).
T = pruebasPerformance; (-> porque la realiza un tester).
T = tuning; (-> porque tarda menos de cuarenta horas).

?- riesgo(T).
T = cacheDistribuida.
T = pruebasPerformance.
```

3. Definir el predicado meFaltanPara/2 que relaciona a una tarea con la lista de todas las tareas que deben ser terminadas antes de poder comenzar con T y no han sido realizadas.
?- meFaltanPara(tuning, Faltantes).
Faltantes = [pruebasPerformance, cacheDistribuida].

4. Usando el predicado forall definir el predicado puedoHacer/1 que indica si una tarea puede ser realizada. (Una tarea puede ser realizada cuando sus precedentes han sido realizadas). El predicado se debe poder usar de la siguiente forma:

```
?- puedoHacer(T).
T = cacheDistribuida;
```
(es la única tarea que puedo hacer en este momento, ya que login está realizada, para pruebasPerformance me falta cacheDistribuida y para tuning me falta pruebasPerformance).

5. Para poder designar horas de trabajo de varios roles distintos a una misma tarea se realiza la siguiente modificación en el hecho tarea/2:
```
tarea(login, [trabajo (80, programador)]).
tarea(cacheDistribuida, [trabajo(40, arquitecto), trabajo(80, diseniador)]).
tarea(pruebasPerformance, [trabajo(100, tester), trabajo(20, analista)]).
tarea(tuning, [trabajo(40, arquitecto), trabajo(20, tester)]).
```

De esta forma por ejemplo para la tarea cacheDistribuida ahora se necesitan 40 horas de un arquitecto y 80 horas de un diseñador.

Se pide definir el predicado muchoTesting/1 que indique si una tarea tiene más de 40 horas de testing, por ejemplo:

```
?- muchoTesting(pruebasPerformance).
Yes.

?- muchoTesting(tuning).
No. (-> menos de 40 horas de testing).

?- muchoTesting(login).
No. (-> no tiene horas de testing planificadas).
```

Ejercicio 5 - dominó

El juego usual del dominó tiene 28 fichas diferentes. Cada ficha es rectangular y tiene grabado en cada extremo un número de puntos entre 0 y 6. P.ej., estas son algunas fichas:

```
| 0 | 1 |
| 1 | 2 |
| 3 | 6 |
```

todas las fichas son distintas entre sí.

Siguendo las reglas del juego, las fichas se colocan formando una cadena de tal manera que cada par de fichas consecutivas tienen iguales números correspondientes a los dos extremos que se tocan. P.ej. esto podría ser el estado del juego en un momento

```
| 0 | 1 || 1 | 4 || 4 | 2 || 2 | 2 || 2 | 5 |
```

Cada jugador posee un conjunto de piezas, las cuales utiliza por turno para ir agregando a la cadena de piezas del juego; puede agregar su ficha en cualquiera de los dos extremos. P.ej. en el ejemplo anterior las siguientes fichas se pueden agregar

```
| 0 | 4 |  (por la izquierda)
| 0 | 6 |  (por la izquierda)
| 5 | 1 |  (por la derecha)
| 5 | 0 |  (por cualquiera de los dos lados)
```

Cuando un jugador no puede agregar ninguna pieza, este debe ceder su turno.

Lo que sigue es parte de la base de conocimiento

```
tieneFicha(carlos,ficha(0,4)).
tieneFicha(carlos,ficha(0,6)).
tieneFicha(carlos,ficha(5,1)).
tieneFicha(german,ficha(5,0)).
tieneFicha(miguel,ficha(3,2)).
tieneFicha(miguel,ficha(3,3)).
tieneFicha(juan,ficha(1,6)).

estado([ficha(0,1),ficha(1,4),ficha(4,2),ficha(2,2),ficha(2,5)]).
```

(en algún momento les va a convenir agregar algunos hechos más).

Escribir las cláusulas que haga falta para resolver el predicado cedeTurno/1, que se verifica para los jugadores que deben ceder su turno. Lograr que este predicado sea inversible.
En el ejemplo, si consulto

```
?- cedeTurno(X)
```

me debe dar las respuestas

```
X = miguel
X = juan
```

Ayuda: vale definir los predicados auxiliares que hagan falta.

Ejercicio 6 - Liga de Fútbol

Hay que desarrollar un programa en Prolog con el que podamos hacer consultas sobre los partidos desarrollados en una liga de fútbol.

Los hechos están dados por:

%fecha(NroFecha, partido(EquipoLocal,GolesLocal,EquipoVisitante,GolesVisitante)).

Por ejemplo

```
fecha(1,partido(mandiyu,0,losAndes,2)).
fecha(1,partido(yupanqui,2,claypole,1)).
fecha(1,partido(jjurquiza,1,cambaceres,4)).
fecha(2,partido(yupanqui,4,jjurquiza,2)).
fecha(2,partido(losAndes,3,claypole,1)).
fecha(2,partido(cambaceres,2,mandiyu,2)).
fecha(3,partido(jjurquiza,3,losAndes,1)).
fecha(3,partido(mandiyu,2,claypole,2)).
fecha(3,partido(cambaceres,3,yupanqui,2)).
```

Tengan en cuenta que no hay partidos que se hagan ida y vuelta en esta liga, o sea, dados dos equipos hay cargado a lo sumo un partido en el cual se enfrentan.
Suponer que la carga de los hechos es correcta, no hacer validaciones.

Desarrollar los siguientes predicados:

a) rival/3, que nos dice quién fue el rival de un equipo en una fecha. Algunas consultas de ejemplo con sus respuestas

```
?- rival(losAndes,2,claypole).
Yes

?- rival(claypole,2,losAndes).
Yes

?- rival(claypole,2,mandiyu).
No

?- rival(claypole,F,mandiyu).
F = 3;
No
```

b) golesLocalFecha/2, que relaciona un número de fecha con el total de goles que hicieron los equipos locales en sus partidos de dicha fecha.

c) golesFecha/2, que relaciona un número de fecha con el total de goles que se hicieron en los partidos de una fecha.

d) jugaron/2 que relaciona dos equipos y nos responde si jugaron entre sí en alguna fecha de la liga.

e) faltaQueJueguen/2 que relaciona a dos equipos si aún no jugaron entre ellos, p.ej. Mandiyú y J.J.Urquiza, en cualquier orden.

f) gano/2 y **perdio/2** que relaciona dos equipos de acuerdo al resultado del partido jugado entre ambos. Un equipo ganó cuando la cantidad de goles que hizo es mayor que la de su rival.

g) empataron/2 que utilice gano/2 y perdio/2

h) Uno que permita conocer los **puntos** que hizo un equipo (se pueden hacer dos predicados auxiliares **partidosGanados/2 partidosEmpatados/2**).
Cada equipo recibirá 3 puntos por partido ganado, 1 por partido empatado y 0 por partido perdido.

i) equipoImprevisible/1 se dice que un equipo es imprevisible si: le ganó a un equipo A, perdió con otro equipo B, pero B a su vez le ganó a A. Este predicado debe ser inversible, vale agregar hechos.

j) Un predicado **goles/2** que relacione un equipo con la cantidad de goles que convirtió en toda la liga

k) Uno que permita saber si un equipo es **invicto** (si gano todos sus partidos) y otro que permita saber si un equipo es **"lucer"** (si perdió todos sus partidos)

l) masCapoQue/2. Mi equipo es más capo que el tuyo si el mío le ganó al tuyo, y también les ganó a todos a los que le ganó el tuyo.

m) campeon/1. Un equipo es campeón si es el que más puntos tiene y el que convirtió más goles

n) Por último uno con el que podamos conocer la **tabla** de posiciones.
La tabla debe ir de mayor a menor cantidad de puntos relacionando cada equipo con sus puntos asociados.

AYUDA: el predicado **setof/3** tiene una funcionalidad igual que la de **findall/3** con la salvedad que el tercer argumento es una lista sin elementos repetidos

Ejercicio 7 – TEG

En vista de que gran parte de las personas (chicos y grandes) abandonan los juegos clasicos por modernos juegos de PC, la jugueteria de Lanus Quindimil SRL, decide llevar su negocio al terreno digital para poder competir con las nuevas formas de espoarcimiento.
Asi es como se comunican con nosotros solicitandonos que realicemos el desarrollo de ciertas consultas para un tablero de TEG, que se realizan cada cierto tiempo.
Cada vez que se realizan las consultas, vamos a contar con ciertos hechos como los que ejemplificamos a continuación.

```
/* distintos paises */
paisContinente(americaDelSur, argentina).
paisContinente(americaDelSur, bolivia).
paisContinente(americaDelSur, brasil).
paisContinente(americaDelSur, chile).
paisContinente(americaDelSur, ecuador).

paisContinente(europa, alemania).
paisContinente(europa, españa).
paisContinente(europa, francia).
paisContinente(europa, inglaterra).

paisContinente(asia, aral).
paisContinente(asia, china).
paisContinente(asia, gobi).
paisContinente(asia, india).
paisContinente(asia, iran).

/*países importantes*/
paisImportante(argentina).
paisImportante(kamchatka).
paisImportante(alemania).

/*países limítrofes*/
limitrofes([argentina,brasil]).
limitrofes([bolivia,brasil]).
limitrofes([bolivia,argentina]).
limitrofes([argentina,chile]).

limitrofes([espana,francia]).
limitrofes([alemania,francia]).

limitrofes([nepal,india]).
limitrofes([china,india]).
limitrofes([nepal,china]).
limitrofes([afganistan,china]).
limitrofes([iran,afganistan]).

/*distribucion en el tablero */
ocupa(argentina, azul, 4).
ocupa(bolivia, rojo, 1).
ocupa(brasil, verde, 4).
ocupa(chile, negro, 3).
ocupa(ecuador, rojo, 2).

ocupa(alemania, azul, 3).
ocupa(españa, azul, 1).
ocupa(francia, azul, 1).
ocupa(inglaterra, azul, 2).
```

```
ocupa(aral, negro, 2).
ocupa(china, verde, 1).
ocupa(gobi, verde, 2).
ocupa(india, rojo, 3).
ocupa(iran, verde, 1).

/*continentes*/
continente(americaDelSur).
continente(europa).
continente(asia).

/*objetivos*/
objetivo(rojo, ocuparContinente(asia)).
objetivo(azul, ocuparPaises([argentina, bolivia, francia, inglaterra, china])).
objetivo(verde, destruirJugador(rojo)).
objetivo(negro, ocuparContinente(europa)).
```

Se solicita construir un programa que permita la resolución de las siquientes consultas.

Todos los predicados deben ser inversibles, salvo aclaración explícita en contrario.

1) **estaEnContinente/2:** Relaciona un jugador y un continente si el jugador ocupa al menos un país en el continente.
2) **cantidadPaises/2:** Relaciona a un jugador con la cantidad de paises que ocupa.
3) **ocupaContinente/2:** Relaciona un jugador y un continente si el jugador ocupa totalmente al continente.
4) **leFaltaMucho/2:** Relaciona a un jugador y un continente si al jugador le falta ocupar más de 2 países de dicho continente.
5) **sonLimitrofes/2:** Relaciona 2 países si son limítrofes.
6) **esGroso/1:** Un jugador es groso si cumple algunas de estas condiciones:
   - ocupa todos los paises importantes,
   - ocupa mas de 10 paises
   - o tiene mas de 50 ejercitos.
7) **estaEnElHorno/1:** un pais esta en el horno si todos sus paises limitrofes estan ocupados por el mismo jugador que no es el mismo que ocupa ese pais.
8) **esCaotico/1:** un continente es caotico si hay mas de tres jugadores en el.
9) **capoCannoniere/1:** es el jugador que tiene ocupado mas paises.
10) **ganadooor/1:** un jugador es ganador si logro su objetivo

Ejercicio 8 – Subtes

Se parte de una descripción de la red de subtes con esta pinta:

```
linea(a,[plazaMayo,peru,lima,congreso,once,castroBarros,primeraJunta]).
linea(c,[retiro,diagNorte,avMayo,moreno,constitucion]).
linea(d,[catedral,nueveJulio,medicina,pzaItalia,carranza]).
linea(h,[rivadavia,belgrano,sanJuan,parquePatricios]).
combinacion(peru,catedral).
combinacion(lima,avMayo).
combinacion(diagNorte,9julio).
combinacion(once,rivadavia).
```

1. Usando estos hechos, definir los siguientes predicados
   a. `pasaPor(Linea,Estacion)`
   b. `combinan(Linea1,Linea2)`
   c. `cantidadEstacionesEntre(Estacion1,Estacion2,Cant)`
      p.ej. entre Lima y Primera Junta hay 4 estaciones.
      Entre estaciones de distintas líneas no tiene que dar ninguna respuesta.
      Ayuda: predicado nth1/3 que viene con SWI Prolog.

d. `esCabecera(Estacion)`
   P.ej. Retiro y Constitucion son cabeceras, Moreno no.
e. `esLineaUniversal(Linea)`
   Una línea se dice "universal" si combina con todas las otras.

2. Ahora definir estos …
   a. `cabeceraMasCercana(Estacion,Cabecera)` , relaciona cada estación con la cabecera que tiene más cerca (según la diferencia de estaciones).
   b. `hayViajeDirecto(Estacion1,Estacion2)` , si puedo viajar sin necesidad de hacer combinación
   c. `hayViajeConCombinacion(Estacion1,Estacion2)` , si puedo viajar haciendo una combinación
   d. `hayViaje(Estacion1,Estacion2)` , si puedo viajar con hasta una combinación
   e. `longitudViaje(Estacion1,Estacion2,Cant)` , cuántas estaciones, si es combinación sumando los dos tramos